AUTOMATYKA • 2005 • Tom 9 • Zeszyt 3

Dominik Sankowski*, Krzysztof Strzecha*, Michał Janicki*

Projekt aplikacji sterującej Thermowet

1. Wprowadzenie

Artykuł obejmuje opis prac nad projektem nowego oprogramowania systemu Thermowet wykonanych w Katedrze Informatyki Stosowanej.

W stosunku do poprzednich, istniejących wersji oprogramowania systemu Thermowet autorzy zauważyli nikłe zastosowanie zaawansowanych technik programowania obiektowo zorientowanego, w szczególności wzorców projektowych [2]. Wersje te charakteryzują się niskim stopniem modularyzacji funkcjonalności całości aplikacji, przez co wyraźnie tracą na swojej przenośności i zdolności akceptowania zmian, zarówno funkcjonalnych, jak i strukturalnych. Ponadto wersje te charakteryzują się wysokim stopniem zależności pomiędzy poszczególnymi modułami składowymi, w szczególności nie uwzględniają warstwy pośredniej pomiędzy elementami kontrolnymi i wykonawczymi. Cechy te są przeciwstawne do zaleceń zaawansowanych metod programowania obiektowo zorientowanego, w szczególności wzorca projektowego MVC (*Model-View-Controller*) [4], zalecającego oddzielenie elementów wykonawczych od elementów sterujących i elementów interfejsu użytkownika – prezentacji wyników. Ponadto zauważalne jest nikłe zastosowanie standardowych klas kolekcji, takich jak listy czy słowniki. Budowie nowej wersji oprogramowania przyświecała idea projektu, który nie posiadałby tych ograniczeń.

2. Zaprojektowane i zaimplementowane moduły

Poniższy opis zawiera charakterystykę poszczególnych zaprojektowanych i zaimplementowanych modułów, wraz ze szczegółami implementacyjnymi.

2.1. Moduł komunikacji szeregowej

Zakres prac projektowych i implementacyjnych w przypadku modułu komunikacji szeregowej obejmował utworzenie diagramu klas całości modułu i określenie ról każdej z klas.

* Katedra Informatyki Stosowanej, Politechnika Łódzka

Każdej z klas przypisane zostały odpowiednie metody i pola realizujące założenia funkcjonalne i strukturalne:

- wysoki stopień enkapsulacji poszczególnych funkcjonalności dla każdej z klas;
- niski stopień wzajemnej zależności pomiędzy poszczególnymi klasami;
- separacja klas odpowiedzialnych za podstawowe operacje portu szeregowego, formatowanie ramki protokołu MODBUS, obliczanie sumy kontrolnej CRC;
- gotowość na zmiany i rozszerzenia o dodatkową funkcjonalność.

Zaprojektowany diagram klas [1] przedstawia rysunek 1. W projekcie zostały wykorzystane wzorce projektowe mostu i fabryki abstrakcyjnej. Uzyskana struktura charakteryzuje się wysokim poziomem enkapsulacji funkcjonalności. Uzyskany projekt posłużył jako szkielet implementacyjny nowej wersji modułu komunikacji szeregowej oprogramowania sterującego systemem Thermowet.

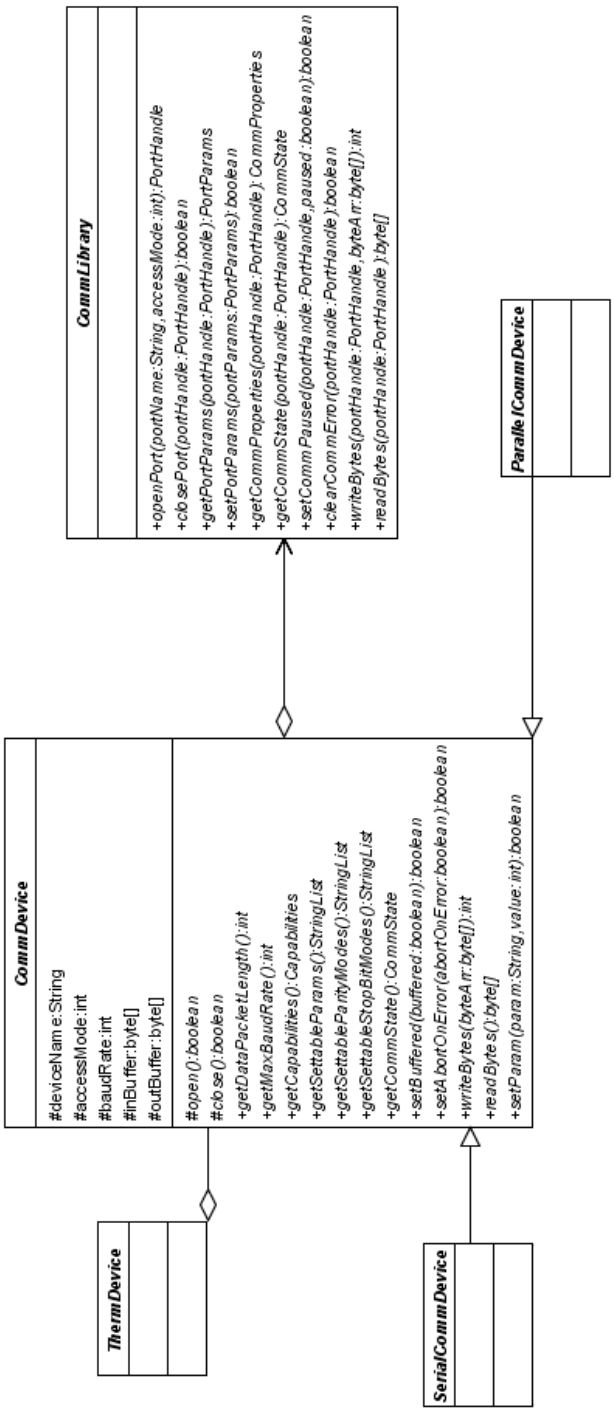
2.2. Moduł pętli eksperymentu i szeregowania zadań

Zakres prac projektowych i implementacyjnych w przypadku modułu pętli eksperymentu i szeregowania zadań objemował zarówno utworzenie diagramu klas całości modułu i określenie ról każdej z klas, jak i jego pełną implementację.

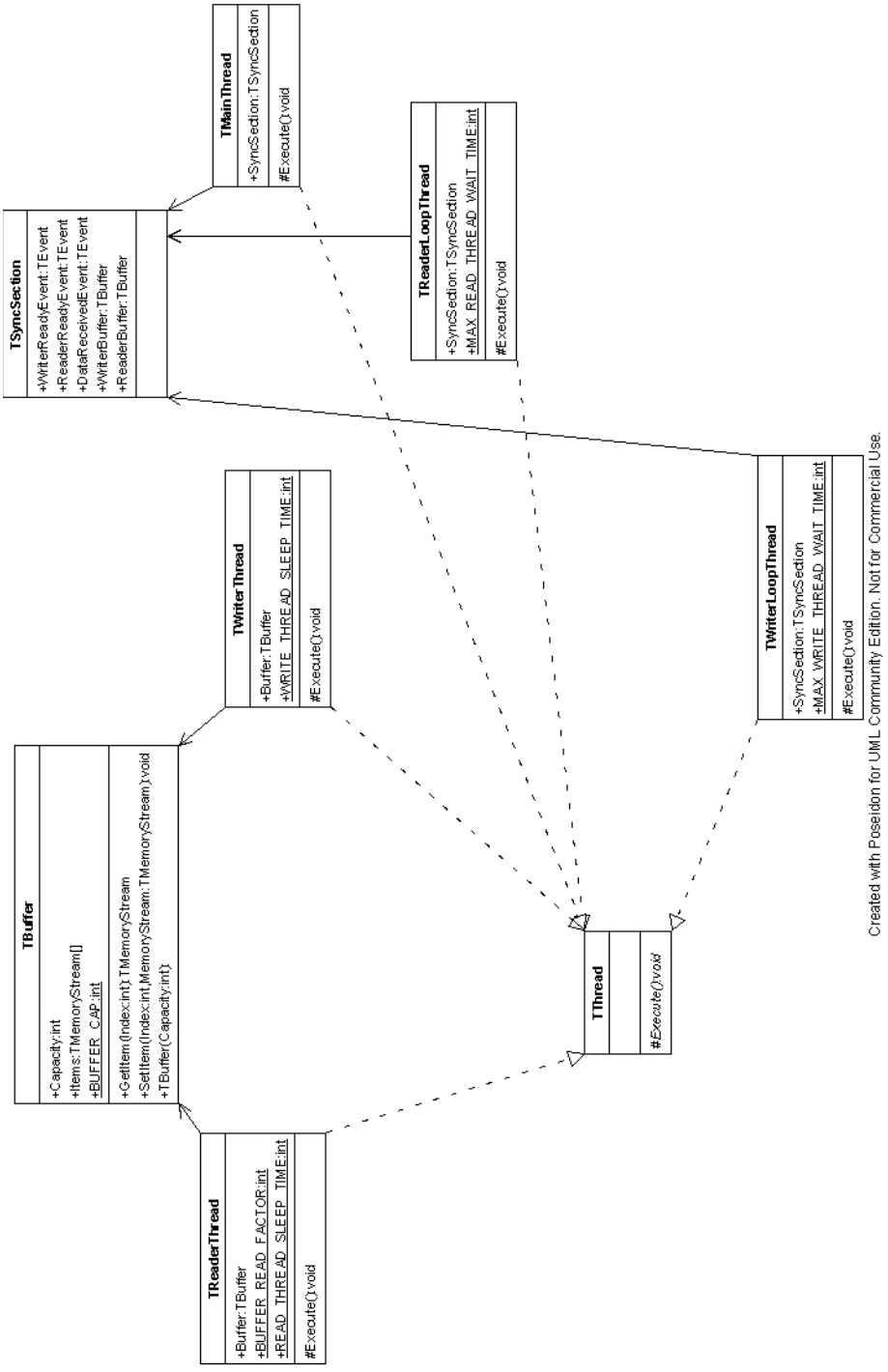
Każdej z klas przypisane zostały odpowiednie metody i pola realizujące założenia funkcjonalne i strukturalne:

- wyłonienie i separacja wątku sczytującego dane z urządzeń zewnętrznych i wątku przetwarzającego uzyskane w ten sposób dane;
- utworzenie struktury bufora pomiarowego, wprowadzenie pojemności maksymalnej bufora i implementacja pomysłu dynamicznego tworzenia i stopniowego wypełniania bufora dla każdej serii pomiarów odpowiadającej pojedynczemu krokowi eksperymentu;
- implementacja synchronizacji pomiędzy poszczególnymi wątkami.

Zaprojektowany diagram klas przedstawia rysunek 2. Uzyskana struktura charakteryzuje się wysokim poziomem enkapsulacji funkcjonalności. Funkcjonalność modułu została zaprojektowana w taki sposób, że z każdym obrotem pętli eksperymentu tworzony jest nowy bufor pomiarowy. Bufor ten jest następnie wypełniany przez wątek sczytujący dane z urządzeń pomiarowych, aż do jego całkowitego zapełnienia określonego przez maksymalną pojemność. W tym czasie wątek przetwarzający jest wstrzymany i oczekuje na nadejście nowego wypełnionego bufora lub – jeśli otrzymał taki bufor wcześniej – zajęty jest jego przetwarzaniem. Wątek sczytujący informuje wątek przetwarzający o dostępności nowego bufora pomiarowego za pośrednictwem sekcji krytycznych. W ten sposób uzyskuje się naprzemienną pracę obu wątków oraz gwarancję poprawności synchronizacji pracy programu. Sekcje krytyczne i zrealizowana w ten sposób wzajemna komunikacja pomiędzy wątkami jest tak skonstruowana, że niemożliwe jest wystąpienie sytuacji, w której pętla pomiarowa zawiesza się. Jest tak również dlatego, że nie ma niebezpieczeństwa przepełnienia bufora danymi pomiarowymi, ponieważ wątki wzajemnie oczekują na siebie i w danej chwili mogą istnieć maksymalnie dwa bufory pomiarowe – jeden aktualnie przetwarzany i jeden aktualnie wypełniany – o ustalonej pojemności maksymalnej.



Rys. 1. Diagram klas modułu komunikacji szeregowej



Rys. 2. Diagram klas modułu pętli eksperymentu i szeregowania zadań

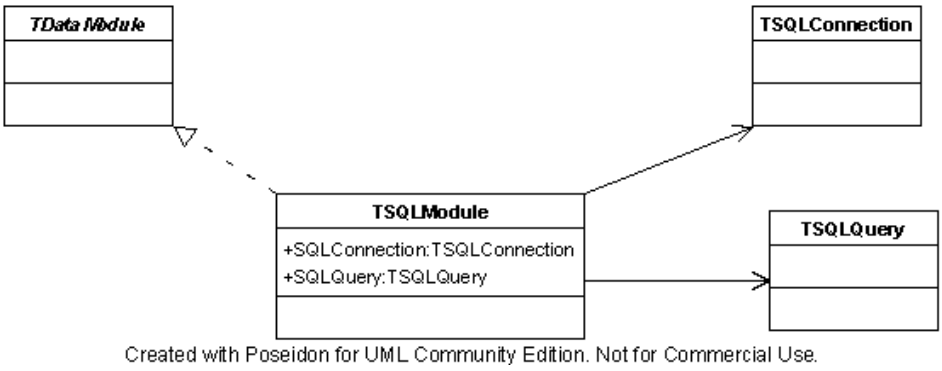
2.3. Moduł dostępu do bazy danych i wykonywania kwerend

Zakres prac projektowych i implementacyjnych w przypadku modułu dostępu do bazy danych i wykonywania kwerend objemował zarówno utworzenie diagramu klas całości modułu i określenie ról każdej z klas, jak i jego pełną implementację.

Każdej z klas przypisane zostały odpowiednie metody i pola realizujące założenia funkcjonalne i strukturalne:

- wyłonienie klasy realizującej fizyczne połączenie z bazą danych, ogólnodostępne dla wszystkich modułów aplikacji;
- wyłonienie klasy realizującej buforowane preparowane zapytania do bazy danych;
- odseparowanie logiki aplikacji od warstwy modelu danych poprzez zastosowanie wzorca MVC.

Zaprojektowany diagram klas przedstawia rysunek 3. Uzyskana struktura charakteryzuje się wysokim poziomem enkapsulacji funkcjonalności. Funkcjonalność modułu została zaprojektowana w taki sposób, że serwer bazy danych może zostać zmieniony w zależności od potrzeb projektu przez zastosowanie odpowiedniego sterownika w standardzie dbExpress. Struktura logiczna modułu jest natomiast całkowicie niezależna od jego doboru. Wykonywanie kwerend odbywa się poprzez dostępny w C++ Builder mechanizm wysokopoziomowego przekazywania parametrów do zapytania preparowanego. Gwarantuje to wraz z odpowiednim doborem sterownika bazy danych pożądaną szybkość wykonywania zapytań.



Rys. 3. Diagram klas modułu dostępu do bazy danych i wykonywania kwerend

2.4. Moduł przetwarzania i analizy obrazu

Zakres prac projektowych i implementacyjnych w przypadku modułu przetwarzania i analizy obrazu objemował zarówno utworzenie diagramu klas całości modułu i określenie ról każdej z klas, jak i jego pełną implementację.

Każdej z klas przypisane zostały odpowiednie metody i pola realizujące założenia funkcjonalne i strukturalne:

- wysoki stopień enkapsulacji poszczególnych funkcjonalności dla każdej z klas;
- łatwość rozszerzenia uzyskanej struktury o dodatkowe algorytmy;
- przenośność klasy przechowującej bitmapy.

Zaprojektowany diagram klas przedstawia rysunek 4. Uzyskana struktura charakteryzuje się wysokim poziomem enkapsulacji funkcjonalności. Funkcjonalność modułu została zaprojektowana w taki sposób, że poszczególne algorytmy są całkowicie od siebie niezależne. Dodatkowo zastosowanie wspólnej warstwy abstrakcji dla każdego z algorytmów, traktowanych jako osobne jednostki wykonawcze, pozwala na łączenie ich działania w łańcuchy bardziej złożonych operacji.

Współautor mgr inż. Michał Janicki zaimplementował algorytmy przetwarzania w oparciu o ich opis słowny zawarty w sprawozdaniu merytorycznym [3]. Każdy z algorytmów został zaimplementowany od podstaw, bez odniesienia do poprzednio istniejących wersji. Każdy z algorytmów został zoptymalizowany i sprawdzony, zarówno pod kątem poprawności, jak i pod kątem modularyzacji całości rozwiązania.

Ponadto współautor zmodyfikował wybrane algorytmy według własnego pomysłu, w szczególności:

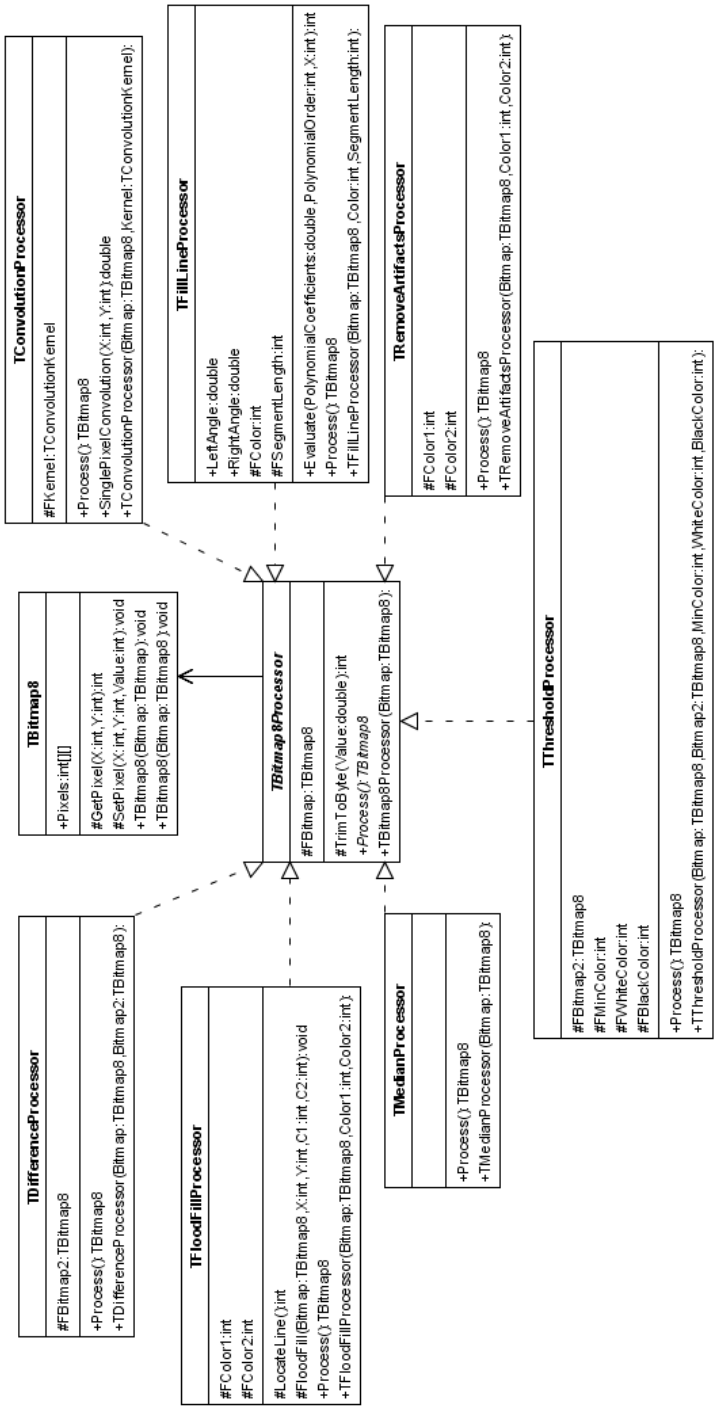
- algorytm „wylewania atramentu”,
- algorytmy aproksymacji wielomianowej.

Pierwszy z wymienionych algorytmów został zaimplementowany bez użycia rekurencji z wylewaniem postępującym zarówno w kierunkach pionowym, jak i w kierunku poziomym. Grupa algorytmów aproksymacji wielomianowej została zmodyfikowana tak, aby stopień wielomianu dobierany był optymalnie w stosunku do miary błędu aproksymacji. Uzyskano w ten sposób dokładniejsze wyniki. Współautor zadbał o odseparowanie klas realizujących funkcjonalność splotu sygnałów dwuwymiarowych, algorytmu eliminacji Gaussa, aproksymacji na zbiorze dyskretnym oraz obliczenia pochodnej wielomianu tak, aby mogły być wykorzystywane niezależnie od całości aplikacji.

Ponadto współautor zaprojektował i zaimplementował klasę przechowującą obrazy w 8-bitowej skali szarości. W klasie została zaimplementowana odpowiednia funkcjonalność konwersji 24-bitowego obrazu kolorowego. Całość została zaprojektowana tak, aby zmaksymalizować łatwość i wydajność współpracy z algorytmami przetwarzania.

2.5. Moduł zbierania obrazu

Zakres prac projektowych i implementacyjnych w przypadku modułu zbierania obrazu objemował zarówno utworzenie diagramu klas całości modułu i określenie ról każdej z klas, jak i jego pełną implementację.



Rys. 4. Diagram klas modułu przetwarzania i analizy obrazów

Każdej z klas przypisane zostały odpowiednie metody i pola realizujące założenia funkcjonalne i strukturalne:

- wysoki stopień enkapsulacji poszczególnych funkcjonalności dla każdej z klas,
- niezależność rozwiązania od doboru sterownika zbierania obrazu,
- integracja rozwiązania z zaimplementowaną klasą przechowującą obrazy w 8-bitowej skali szarości.

Uzyskana struktura charakteryzuje się wysokim poziomem enkapsulacji funkcjonalności. Funkcjonalność modułu została zaprojektowana w taki sposób, że urządzenie zbierania obrazu może zostać zmienione w zależności od potrzeb projektu poprzez dziedziczenie z odpowiedniej klasy warstwy abstrakcji.

2.6. Moduł interfejsu użytkownika i wzorzec MVC

Zakres prac projektowych i implementacyjnych w przypadku modułu interfejsu użytkownika objemował zarówno utworzenie diagramu klas całości modułu i określenie ról każdej z klas, jak i jego pełną implementację.

Każdej z klas przypisane zostały odpowiednie metody i pola realizujące założenia funkcjonalne i strukturalne:

- wysoki stopień enkapsulacji poszczególnych funkcjonalności dla każdej z klas,
- realizacja wzorca MVC, separującego warstwy przetwarzania, modelu danych i prezentacji danych.

Uzyskana struktura charakteryzuje się wysokim poziomem enkapsulacji funkcjonalności. Funkcjonalność modułu została zaprojektowana na bazie istniejących w systemie C++ Builder klas separujących warstwy przetwarzania, modelu danych i prezentacji danych [5] poprzez wprowadzenie pojęcia klasy współdzielonej akcji.

3. Podsumowanie

W sprawozdaniu autorzy wyszczególnili i opisali wykonane prace nad nowym oprogramowaniem sterującym systemem Thermowet.

Literatura

[1] Booch G., Rumbaugh J., Jacobson I.: *The Unified Modeling Language User Guide.* Massachusetts, Addison-Wesley 1998

[2] Gamma E., Helm R., Johnson R., Vlissides J.: *Design Patterns: Elements of Reusable Object-Oriented Software.* Massachusetts, Addison-Wesley 1995

[3] Sankowski D.: *Automatyczne pomiary napięcia powierzchniowego i kąta zwilżania materiałów w wysokich temperaturach.* Katedra Informatyki Stosowanej, t. 1, 2000, 61–104 (Sprawozdanie merytoryczne)

[4] Stelting S.: *Applied Java Patterns.* California, Prentice Hall PTR 2001

[5] Swart B.: *Borland C++ Builder Developer's Guide.* Indiana, Sams Publishing 2003